Journal of High School Science

Hotel reservation cancellations: predictive modeling and feature impact analysis

Lau G[1], Kerimov A[2]

Submitted: October 31, 2024, Revised: version 1, February 2, 2025, version 2, March 5, 2025, version 3, March 26, 2025
Accepted: March 27, 2025

Abstract

Knowing if and when a reservation will be canceled is crucial to hotels. They can try reselling rooms that are predicted to be canceled to minimize revenue losses and optimize inventory. We sought to use machine learning models to predict hotel reservation cancellations with higher f1-scores than previous studies because of their crucial impact on hotels. Using our best model of a hyper-tuned random forest model, we first used all available features in our data. We achieved test f1-scores of 0.92, for not canceled reservations and 0.83 for canceled bookings. We also performed feature importance analysis to understand the marginal contribution of each feature on the model's predictions. As a result, we identified that the top ten features were the most important. We experimented with a tuned random forest using only the top four and top ten features and found a model trained on only the top ten features performed competitively with our best-performing model. This model achieved test f1-score of 0.91 for not canceled reservations and 0.82 for canceled bookings. The model trained on the top four most contributing features achieved a testing f1-score of 0.89 for not canceled reservations and 0.78 for canceled bookings. Compared to our best-performing model, this model, trained on the top four features, presented with a maximum 5% difference for canceled bookings. As a result of this study, a model trained on the top four features achieved acceptable trade offs in f1 scores of 0.2 and 0.4 for not canceled and canceled reservations respectively.

Keywords

Hotel reservation cancellation, Machine Learning, Random Forest, XGBoost, Hyperparameter tuning, SHAP Analysis, SMOTE, Finance, Overbooked

[1]Corresponding author: Grant Lau, Aragon High School, 900 Alameda de las Pulgas, San Mateo, CA 94402, USA. glau6002@gmail.com

[2]Abdulla Kerimov, Stanford Rock Physics Lab, Stanford University, 450 Jane Stanford Way, Stanford, CA 94305, USA. akerimov@alumni.stanford.edu

https://doi.org/10.64336/001c.133741

Introduction

Customer reservations are the basis against which a hotel's revenue is measured. Predicting reservation cancellations is crucial to hotels. If a hotel recognizes cancellations are likely, they can try overbooking rooms to maximize revenue. This information is also helpful for hotels as they can use it for marketing and analysis.

Hotels can efficiently predict booking cancellations by using machine learning models. Machine learning is an effective way to approach this challenge as each booking contains many informational features on which to train a model. Machine learning has already been applied to other industries with great success (1, 2). In fact, some studies have already used machine learning models to try to predict hotel reservation cancellations.

Antonio et al. (3) developed a model that was trained on data from two different hotels in Portugal. For canceled reservations, their model resulted in a precision scores, sensitivity and accuracy respectively of 0.811, 0.603 and 0.842 for hotel one and 0.869, 0.779 and 0.849 for hotel two. More recently, Chen et al. (4) proposed using a combined Bayesian network-lasso regression model, also trained on data from two Portuguese hotels. Their approach resulted in a lower performance in predicting canceled bookings. Their model had an accuracy score of 0.8186, and significantly lower precision and recall (sensitivity) scores of 0.7597 and 0.5097, respectively.

Recognizing the room for improvement, we sought to construct a better-performing machine learning model to predict hotel booking cancellations. During this process, we also sought to understand how different features in our data affected our models' predictions of the booking status. Using this information, we aimed to maximize the precision and recall of our machine-learning models, while using only limited, most important contributing factors. Finally, we investigated the effect of data imbalance on model performance and the use of various sampling techniques to minimize errors arising from this imbalance.

Dataset

Our dataset consisted of 36,275 hotel bookings (5). Each booking included the booking status, number of adults and children, cancellation history of the party that made the reservation, number of week and weekend nights booked, type of meal plan, if a parking space was required, type of room, lead time, arrival month, arrival day of the week, web platform the booking was made on, average price per room, if they were a repeated guest, and the number of special requests made. Additionally, each reservation could be identified by a booking ID.

The target was the booking status, with the goal of predicting whether the booking party would cancel their hotel reservation. Within the dataset, there were 24,390 reservations that had not been canceled and 11,885 reservations with a canceled status. Hence, the dataset was imbalanced.

Exploratory Data Analysis (EDA)

This section explores and discusses insights and trends in our dataset. While preprocessing our dataset, we performed exploratory data analysis to identify potential trends in the data. Our goal was to understand the relationship

between numeric (eg. the number of adults, lead time, etc) and categorical (eg. repeated guest, arrival month, etc) features in the data and their impact on the target: canceled or not canceled bookings. Figure 1 illustrates the count distribution of canceled and not canceled hotel bookings. There are almost twice as many "not_canceled" reservations as "canceled" ones, meaning our data is imbalanced and will likely be biased towards "not_canceled" reservations.

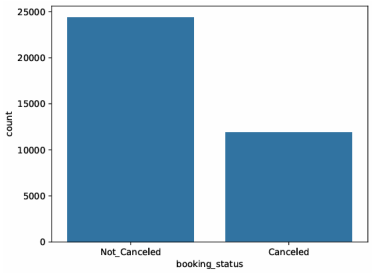

Figure 1. Count distribution of canceled and not canceled bookings

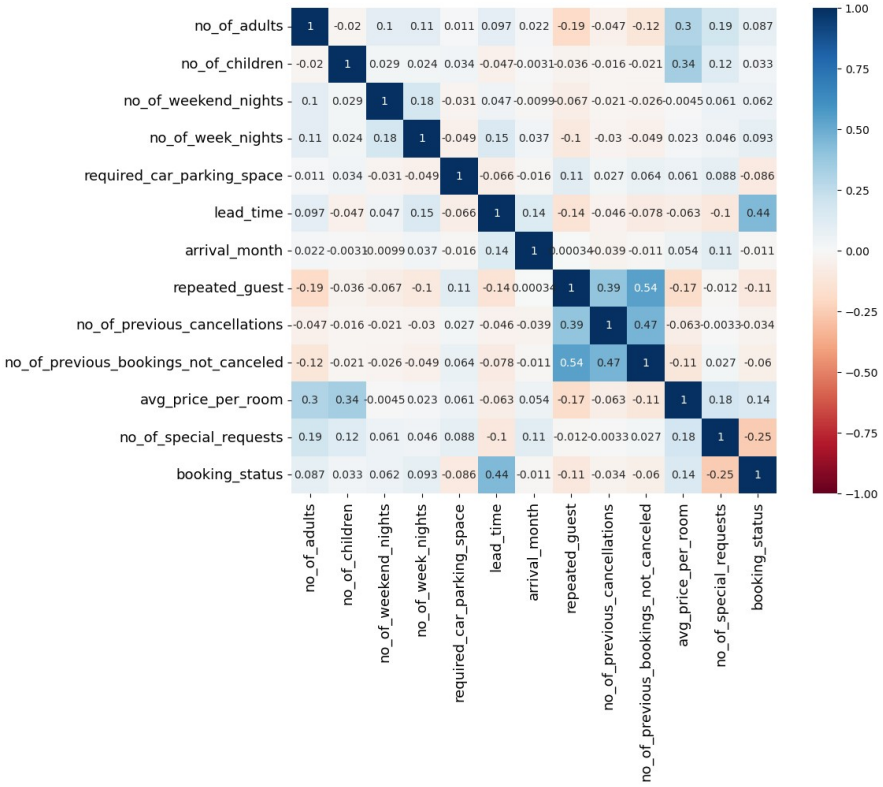

Figure 2. Pearson correlation matrix

We used Pearson correlation to calculate the correlation coefficients between the variables. As seen in Figure 2, 0.54 and 0.47 are the largest correlation coefficients between the number of previous bookings not canceled and repeated guests pair, and the number of

previous bookings canceled and not canceled pairs, respectively. Since there occurred no highly correlated variables with coefficients larger than +/-0.8, we concluded that there was no evidence of collinearity present in the dataset.

The majority of the reservations had two adults registered as shown in Figure 3. Reservations with only one adult had the second highest count, followed by three adults. Regardless of the number of adults on the reservations, in every instance, there were more reservations with a "Not_Canceled" booking status than those with a "Canceled" status.

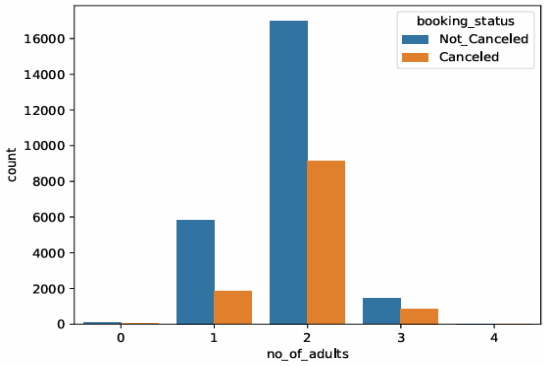

Figure 3. Count distribution of the number of adults per canceled and not canceled bookings

In contrast to the number of adults per reservation, reservations with zero children had the highest count as shown in Figure 4. Bookings with one and two children were significantly lower, with two having the smallest amount of occurrences. Similar to Figure 3, the trend that occurred when examining bookings based on the number of adults was that the number of non-canceled reservations always exceeded the number of canceled reservations.

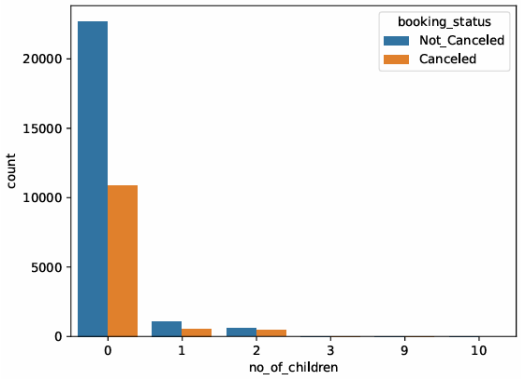

Figure 4. Count distribution of the number of children per canceled and not canceled bookings

All of the bookings in the dataset marked whether or not, they were made by a repeated guest. The number of non-repeat guests significantly outnumbered the amount of repeated guests. While reservations made by non-repeat guests were split between

"canceled" and "not_canceled" status, more bookings had a "not_canceled" status as shown in Figure 5. In comparison to repeat guests, however, none of the repeated guest bookings showed a "canceled" status.

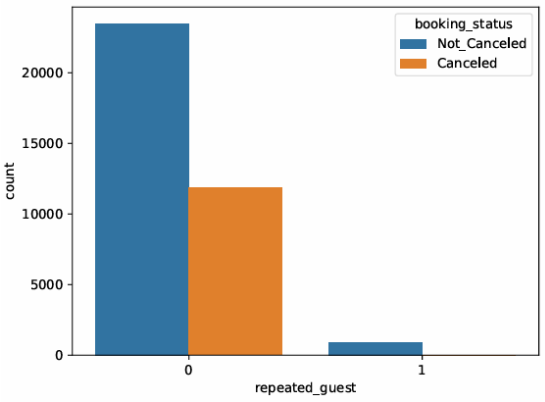

Figure 5. Count distribution of canceled and not canceled bookings with respect to repeating (1) and not repeating (0) guests

Figure 6 shows that the number of reservations and the number of special requests per reservation share an inverse relationship. As the number of special requests increases, the amount of bookings in that number grouping decreases. The majority of the hotel reservations presented with zero special requests. In addition, the amount of not canceled reservations outnumbered the amount of canceled reservations on all accounts.

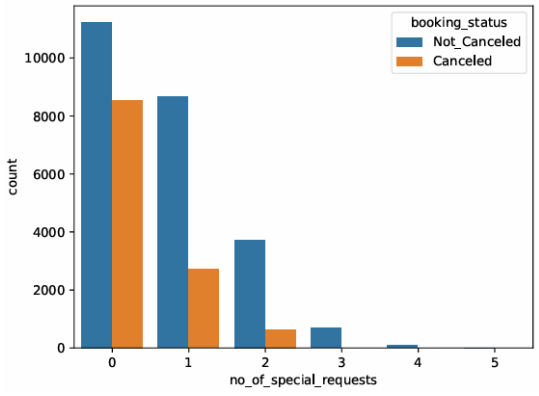

Figure 6. Count distribution of the number of special requests per canceled and not canceled bookings

Lead time is the amount of time prior to a party's stay that they book a reservation. The bookings in our dataset showcased a relationship between lead time and booking status as shown in Figure 7. Interestingly, more reservations with higher median lead times were canceled compared to bookings with lower lead times.

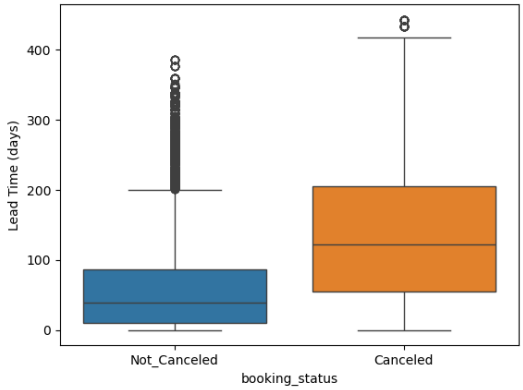

Figure 7. Box plot of canceled and not canceled bookings with respect to lead time

Finally, the average price per room begins increasing in March, peaks in August and September, and starts falling in October as presented in Figure 8. Furthermore, there is a significant average price gap between canceled and not canceled reservations in March. The rise in average price per room during these months is not surprising as many people take vacations during Spring break and the summer months.

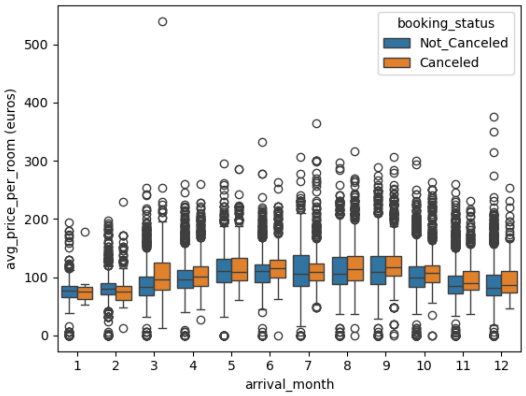

Figure 8. Box plot of the average price per room for given arrival months colored by canceled and not canceled bookings

Methods and Models

Input features and output target

The input features were a mix of numeric and categorical data as shown in Table 1. These input features included the number of adults and children, the number of week and weekend nights, the type of meal plan, if the hotel required a parking space, the type of room reserved, lead time, the arrival month, the day of week arrived, market segment type, if they were a repeat guest, number of previous bookings that were not canceled, the average price per room, and number of special requests. The target is the booking status, predicting whether reservations will be canceled or not canceled.

Table 1. Input features and output target type and description

Variable	Type	Description
no_of_adults	Numerical	Number of adults
no_of_children	Numerical	Number of children
no_of_week_nights	Numerical	Number of weeknights
no_of_weekend_nights	Numerical	Number of weekend nights
type_of_meal_plan	Categorical	Plan 1, Plan 2, Plan 3, Not selected
required_car_parking_ space	Binary	0 = No, 1 = Yes
room_type_reserved	Categorical	Room_type 1, Room_type 2, Room_type 3, Room_type 4, Room_type 5, Room_type 6, Room_type 7
lead_time	Numerical	Days leading up to reservation check-in
arrival_month	Numerical	1 = January, 2 = February, 3 = March, 4 = April, 5 = May, 6 = June, 7 = July, 8 = August, 9 = September, 10 = October, 11 = November, 12 = December
day_of_week	Categorical	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
market_segment_type	Categorical	Aviation, Complementary, Corporate, Offline, Online
repeated_guest	Binary	0 = No, 1= Yes
no_of_previous_ cancellations	Numerical	Number of previous bookings canceled
no_of_previous_bookings_not_canceled	Numerical	Number of previous bookings not canceled
avg_price_per_room	Numerical	Average price per room
no_of_special_requests	Numerical	Number of special requests
booking_status	Categorical	0 = Not_canceled, 1 = canceled

Data encoding

In order to feed the categorical features in our data to our models, we first needed to encode them. Data encoding is the process of transforming text and other non-numerical data values into numbers a model can understand. We chose one-hot encoding (6). This method of data encoding creates a binary column for each textual option within each categorical feature column. Then, it sets the present category's column with a 1 and the rest with a 0. To better explain, an example follows with one of the categorical feature columns in our dataset.

We one-hot encoded "room_type_reserved", "market_segment_type", "type_of_meal_plan", and "day_of_week." The subcategories created for each column following one-hot encoding can be seen in Table 2. As an example, we use the "room_type_reserved" feature. Each reservation has seven different room options when booking the stay and is initially encoded using text. Let us assume that the first

reservation books room type one. After one-hot encoding, the "room_type_reserved" column is expanded into seven different columns, one for each option, and a 1 is placed in the column for room type one. The other six columns will then be set with a 0. This process is performed on all the bookings and for all the categorical feature columns.

Finally, the last column we encoded was the booking status column. Initially, each reservation is marked with a booking status of canceled or not canceled. Because this is textual data, it needed to be encoded for our model. However, instead of one-hot encoding, we used ordinal label encoding (6) where any occurrence of "canceled" was simply replaced with a 0 and instances of "not_canceled" with a 1.

Table 2. Categorical features and their subcategories

Categorical Feature	Subcategories
day_of_week	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
room_type_reserved	Room_type 1, Room_type 2, Room_type 3, Room_type 4, Room_type 5, Room_type 6, Room_type 7
market_segment_type	Aviation, Complementary, Corporate, Offline, Online
type_of_meal_plan	Plan 1, Plan 2, Plan 3, Not selected

Baseline models and evaluation metrics

Before model development, we separated the data into an 80-20 train-test split. Additionally, because our dataset was imbalanced, we set stratify equal to "true." This kept the ratio of non-canceled and canceled reservations the same across both the training and testing data groups. First, we tested a logistic regression model. Specifically, we used a binary logistic regression model because our outcome fell into two categories: canceled or not canceled. Logistic regression uses a logistic or sigmoid function to calculate the probability of an event happening (eg. canceled reservation) and converts probabilities into values between zero and one (7). The model then starts comparing probabilities to the threshold value. A threshold value is established by a user and used to decide which category to place a value. By default, the threshold value is set to 0.5 as it lies midway between 0 and 1. However, this can be adjusted to optimize certain evaluation metrics. If the probability exceeds the threshold, the model predicts that the output's category will be 1. If the probability is lower than the threshold, the model predicts the output's category will be 0.

We then tested a random forest model. A random forest can be thought of as a group of decision trees that work together before making a prediction (8). A decision tree is a series of branches and nodes that act as multiple if-else statements to help the model reach a prediction

(9). Each tree is trained on the data slightly differently than the rest. Given an input, each tree in the group makes a prediction and the one with the highest frequency becomes the random forest's final prediction.

Lastly, we used an eXtreme Gradient Boosting (XGBoost) model. An XGBoost model is the last decision tree in a series of decision trees trained on the mistakes of the one prior (10). When an XGBoost model is initialized, a decision tree is trained and tested on a dataset. Subsequently, its errors are identified and a new decision tree is created and trained on the previous decision tree's errors in addition to the original data. This process is repeated a specified amount of times, making the last tree in the series the designated model for predictions.

To evaluate all of our models, we focused on maximizing the weighted f1-score (6). We prioritized these metrics because they focus on the positive predictions made by the model. A positive prediction in our case is that someone will cancel their hotel reservation. Precision calculates how many positive predictions a model correctly predicted, while recall measures the number of actual positive cases the model predicted as positive. The f1-score is a metric calculated using both precision and recall. Alongside these metrics, we also used confusion matrices to visualize our models' predictions and to understand the degree to which they were overfitting.

Hyperparameter tuning

Once we had finished testing our baseline models, we performed hyperparameter tuning to try to improve our models' performance using a grid search method (11). Hyperparameter tuning is the process of adjusting the parameters of a model to maximize the chosen evaluation metric (6). Changes in various hyperparameters can affect the performance of a model.

For our random forest, we chose to modify the n_estimators, max_features, and min_samples_split hyperparameters. Initially, we set n_estimators to either 500 or 1000; max_features within the range of 4 to 25, incrementing by 2; and min_samples_split to be within the range of 50 to 150, incrementing by 10. To test all these hyperparameters, we performed a grid search algorithm that created all possible combinations and tested our random forest using each combination, reporting the best-performing one.

To hyperparameter tune our XGBoost model, we chose the following hyperparameters: eta, colsample_by_node, subsample, and n_estimators. For our first grid search, we set eta to be within a range of 0.05 to 1, increasing by 0.1; colsample_bytree within 0.2 and 1, increasing by 0.1; subsample between 0.5 and 1, incrementing by 0.1; and n_estimators to 500 and 1000. When hyperparameter tuning all our models using the train data and grid search method, we set cross-validation to 3 and the evaluation metric to weighted f1-score.

Processing the imbalanced dataset

As mentioned previously, our dataset consisted of 24,390 non-canceled reservations and 11,885 canceled reservations, making our data imbalanced. To correct the imbalance in our dataset, we use two different approaches: setting our models' class weight parameter to "balanced" and synthetic minority oversampling technique (SMOTE).

To reduce any bias our models may have had toward (the majority class of) non-canceled reservations, we used the class weight parameter to adjust the importance of each in the data. With our dataset, we increased the importance of the minority class—canceled reservations—to ensure our models paid more attention to it during training. By setting our models' class weight parameter to "balanced", the y-target values in our data are used to calculate the class weights which are inversely proportional to the class counts in the data (12).

When using SMOTE on a dataset, there are two options: oversampling and undersampling (13). Oversampling is the process of synthetically increasing the minority class to be equal in amount to the majority class. As seen in Figure 9, the green circles represent the minority class, and the blue squares depict the majority class. The way SMOTE increases the minority class is by selecting an instance in the minority class and identifying its k nearest neighbors. Then a random neighbor is selected and a new point is synthetically created along the path between the original instance and its neighbor (red circles). This process is repeated until the minority class is equal in size to the majority class. In contrast to oversampling, undersampling is the process of randomly selecting and removing instances in the majority class until they are equal in size to the minority class.

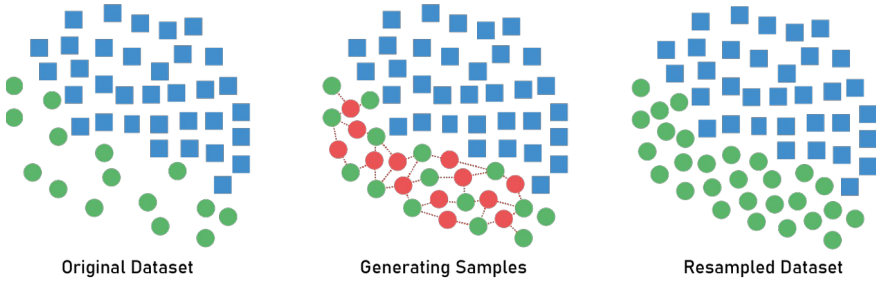

Figure 9. Illustration of the Synthetic Minority Oversampling Technique (14)

Feature importance analysis

To understand the importance of features and their impact on model prediction we used SHapley Additive exPlanations (SHAP). SHAP analysis and Shapley values were originally used as a solution to cooperative game theory —a group of players working together to win a reward need to determine how to divide it fairly among themselves based on their contributions. It was later adapted for machine learning in 2010 to explain machine learning model predictions (15). By using SHAP analysis, we can better understand why our models made certain predictions and what influenced them to do so. At a specific data point, a specific feature's influence on the model's prediction is represented by a numeric value assigned by SHAP values. Then all possible combinations of features and their contributions are used to create a comprehensive understanding of the importance of different features, which can be illustrated using SHAP plots. As a result of SHAP analysis, we identified the most

important and top contributing features to the model prediction.

Results and Discussion

Our models predicted whether a booking, such as a repeat customer with 2 adults and one child, booked for 2 weekday nights in June, with no previous cancellations, 4 days of lead booking time, and a required parking space, would – or would not - be canceled. Note that since this is a classification problem, our models could output the probability of the booking cancellation as well.

For our first approach, logistic regression, Table 3 shows that the train and test f1-scores for non-canceled bookings are 0.86 for both. Additionally, for canceled bookings, logistic regression has an f1-score of 0.67 for training and 0.68 for testing. These consistent f1-scores implied that our model was not overfitting, but performed significantly lower for canceled bookings with an overall low performance. Next, the baseline random forest achieved a training f1-score of 0.99 for both categories, but testing f1-scores of 0.91 for non-canceled and 0.81 for the canceled categories, respectively (Table 3). This large difference between training and testing f1-scores may be indicative of overfitting. The baseline XGBoost model achieved training f1-scores of 0.93 for the non-canceled category and 0.85 for the canceled category, as well as 0.91 and 0.81 testing f1-scores for these categories respectively (Table 3). This model hence seemed to demonstrate some overfitting, but less than that obtained in the baseline random forest model.

We hyper-tuned a random forest with the class weight parameter set to "balanced." With tuned max features of 14, 50 min_samples_split, and 1000 n_estimators, it achieved the best f1-score —train and test of 0.93 and 0.92, respectively, for non-canceled reservations and 0.86 and 0.83, respectively, for canceled bookings.

For processing the imbalanced dataset, we implemented SMOTE on only training data. Using oversampled SMOTE data, tuned max_features set to 18, min_samples_split set to 50, and 1000 n_estimators, our hyper-tuned random forest achieved f1-scores of 0.92 for both categories in training, and testing f1-scores of 0.91 and 0.82 for non-canceled, and canceled reservations, respectively. By switching to undersampled SMOTE data and tuned max_features of 20, our hyper-tuned random forest achieved train and test f-scores of 0.90 and 0.91 for non-canceled bookings, respectively, and 0.89 and 0.82 for canceled reservations, respectively (Table 3).

Finally, we hyper-tuned an XGBoost model on both oversampled and undersampled SMOTE data. Oversampled achieved its best f1-scores with colsample_bytree set to 0.7, eta at 0.15, 500 n_estimators, and subsample set to 0.9. These hyperparameters resulted in training f1-scores of 0.96 for both categories and testing f1-scores of 0.92 for non-canceled bookings and 0.84 for canceled bookings. While our undersampled hyper-tuned XGBoost had colsample_bytree set to 0.90 and subsample set to 0.89, it achieved almost identical f1-scores as its oversampled counterpart.

Table 3. Train and test F1 evaluation scores with best model parameters using different approaches

Approach	Best parameters	Train-test set	F1-Score booking_status 'not_canceled'	F1-Score booking_status 'canceled'
Logistic Regression	Default	Train	0.86	0.67
		Test	0.86	0.68
Baseline Random Forest	Default	Train	0.99	0.99
		Test	0.91	0.81
Baseline XGBoost	Default	Train	0.93	0.85
		Test	0.91	0.81
Tuned random forest + class_weight: 'balanced'	'max_features': 14, 'min_samples_split': 50, 'n_estimators': 1000	Train	0.93	0.86
		Test	0.92	0.83
Tuned random forest + SMOTE Oversampling	'max_features': 18, 'min_samples_split': 50, 'n_estimators': 1000	Train	0.92	0.92
		Test	0.91	0.82
Tuned random forest + SMOTE Undersampling	'max_features': 20, 'min_samples_split': 50, 'n_estimators': 1000	Train	0.90	0.89
		Test	0.91	0.82
Tuned XGBoost + SMOTE Oversampling	'colsample_bytree': 0.7, 'eta': 0.15, 'n_estimators': 500, 'subsample': 0.9	Train	0.96	0.96
		Test	0.92	0.84
Tuned XGBoost + SMOTE Undersampling	'colsample_bytree': 0.90, 'eta': 0.15, 'n_estimators': 500, 'subsample': 0.89	Train	0.97	0.97
		Test	0.91	0.83

The hyper-tuned random forest with tuned max features of 14, 50 min_samples_split, 1000 n_estimators, and class weight parameter set to "balanced" was our best-performing model. Though the "Tuned XGBoost + SMOTE Oversampling" model demonstrated a similarly competitive performance, we ultimately selected the "Tuned Random Forest + Class_Weight: 'Balanced'" model due to its lower variance between train and test F1-scores. For example, "Tuned Random Forest + Class_Weight: 'Balanced'" yielded a training score of 0.86 and a test score of 0.83 for canceled bookings, while "Tuned XGBoost + SMOTE Oversampling" yielded a training score of 0.96 and a test score of 0.84. This suggested that the Random Forest model was less prone to overfitting compared to the XGBoost model, which exhibited a larger discrepancy between training and testing performance. To summarize the performance of our model, we plotted confusion matrices for training and testing data as shown in Figure 10. On the axes of the matrices, 0 represents non-canceled and 1 represents canceled bookings. The top left square represents true negative

values, which are hotel bookings our model correctly predicted as not canceled, and the bottom left represents false negatives, bookings incorrectly predicted as not canceled. The bottom right square symbolizes true positive values, bookings correctly predicted as canceled, and the top right represents false positives, bookings incorrectly predicted as canceled. In Figure 10, the top left and bottom right squares have the most values, demonstrating the model's overall quality performance. The true positive, true negative, false positive, and false negative values are used to calculate a model's precision and recall metrics, as discussed earlier.

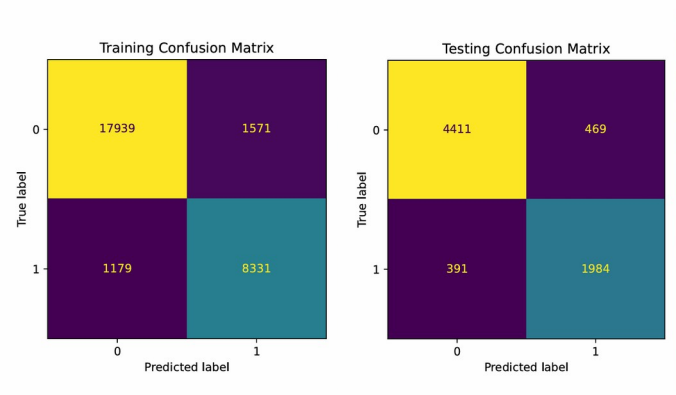

Figure 10. Training and testing confusion matrices of best-performing tuned random forest model ('class_weight': 'balanced', 'max_features': 14, 'min_samples_split': 50, 'n_estimators': 1000)

As seen in Table 4, the best-performing random forest model's ('class_weight': 'balanced', 'max_features': 14, 'min_samples_split': 50, 'n_estimators': 1000) train and test precision scores presented with a difference of 0.02 for both the non-canceled and canceled bookings. Furthermore, its train and test recall and f1-scores differed by 0.01 for non-canceled reservations. Finally, for canceled bookings, the model's train and test scores presented with a difference of 0.04 for recall and a difference of 0.03 for f1-scores. Compared to the other models tested, this model exhibited some of the highest as well as the most consistent scores, making it the best.

Table 4. Classification report of best-performing tuned random forest model ('class_weight': 'balanced', 'max_features': 14, 'min_samples_split': 50, 'n_estimators': 1000)

Status	Train Precision	Test Precision	Train Recall	Test Recall	Train F1-Score	Test F1-Score
'not_canceled'	0.94	0.92	0.92	0.91	0.93	0.92
'canceled'	0.84	0.82	0.88	0.84	0.86	0.83

As mentioned earlier, we performed SHAP analysis on our data to understand the impact of various features on our best-performing (Table 4) model's predictions. We visualized these impacts with beeswarm plots as shown in Figure 11. On the left, is the beeswarm plot for not canceled reservations, and on the right is that for canceled bookings. As seen on the x-axis of the beeswarm plots, positive SHAP values signal a positive impact and negative values represent a negative impact on predictions. We see that factors that positively influence not canceled bookings have an inverse effect on canceled reservations. For example, higher lead time had a negative impact on not canceled bookings, while it positively affected canceled bookings. Furthermore, a higher number of special requests had a positive impact on not canceled bookings, but negatively impacted canceled reservations.

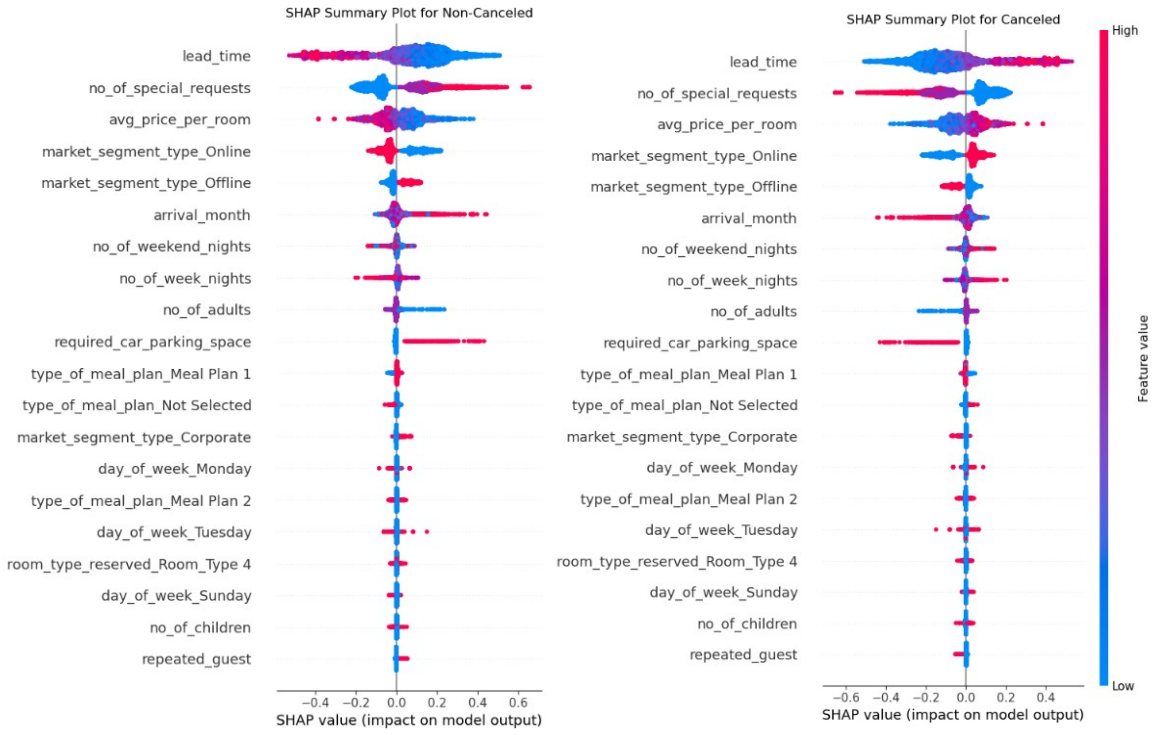

Figure 11. SHAP beeswarm plots displaying the positive and negative effects of input features on not canceled (left) and canceled (right) reservation predictions

Using a bar plot, the overall importance of all the features in making a prediction, is shown in Figure 12. The numbers on the x-axis symbolize the overall percent impact of each feature. It can be seen that lead time, the number of special requests, the average price per room, making the reservation online and offline, the arrival month, the number of week and weekend nights, and the number of adults had the most significant impact on our model's predictions. For example, the lead time had a 17.5% overall impact, the largest of all the features, followed by the number of special requests with ~ 15% overall impact.

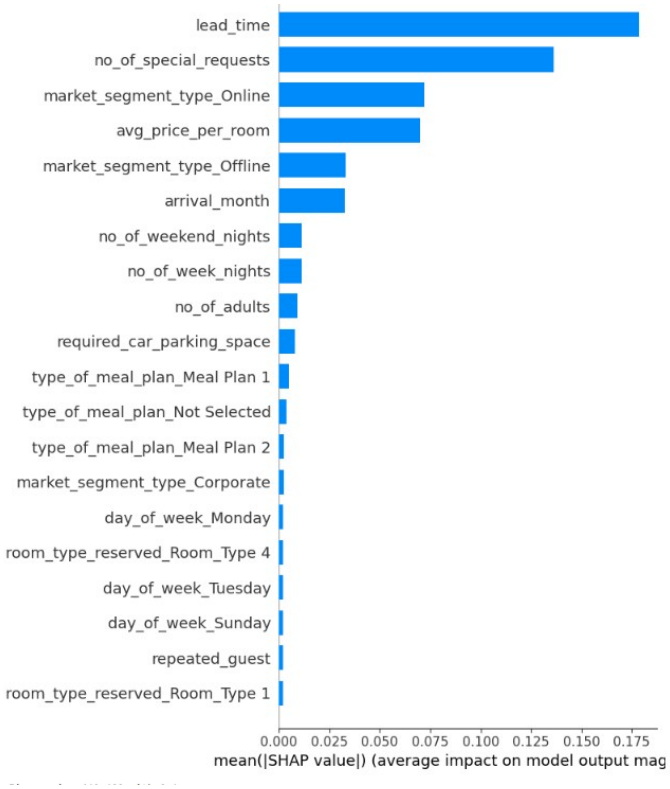

Figure 12. SHAP bar plot displaying the overall impact of input features on model prediction

To test the impact of our top features, we attempted to replicate our best model's performance by using the top four and the top ten features. We first split the data into train and test data with the appropriate features. We then hyperparameter-tuned a model using a grid search using a random forest model with class_weight set to "balanced." The top ten input features were lead time, the number of special requests, the average price per room, making the reservation online and offline, the arrival month, the number of week and weekend nights, and the number of adults as shown in Figure 12.

After hyperparameter tuning, we found the best parameters for our model trained on the top four features max feature of one, min_samples_split of 50, and n_estimators of 1000. This model's train f1-scores were 0.91 for not canceled and 0.82 for canceled bookings. Using the testing data, it scored 0.89 for not canceled and 0.79 for canceled reservations, as shown in Table 5. The model trained on the top ten features achieved its highest scores with max_features of 4, min_samples_split of 50, and n_estimators of 1000. This model's train and test f1-scores were 0.92 and 0.91, respectively, for not canceled bookings, and 0.85 and 0.82, respectively, for canceled reservations as shown in Table 5.

Using the top four and ten features, we produced performances comparable to our best-performing model. The model with only the top

ten features achieved close results to our best model and outperformed all the models in Table 3 in terms of consistency and overall performance, demonstrating that only the top ten features in our data were necessary to achieve competitive performance. The model trained on only four features achieved a maximum 5% difference in testing f1-scores for canceled bookings when compared to our best-performing model. Hence the trade off between performance and computing time may be optimum for this model.

Table 5. Train and test F1 evaluation scores with best model parameters using all, top four and top ten, contributing features

Approach	Best parameters	Train/test set	F1-Score booking_status 'not_canceled'	F1-Score booking_status 'canceled'
Tuned random forest + class_weight: 'balanced' + All features	'max_features': 14, 'min_samples_split': 50, 'n_estimators': 1000	Train	0.93	0.86
		Test	0.92	0.83
Hypertuned RF + class weight + Top **4** features	'max_features': 1, 'min_samples_split': 50, 'n_estimators': 1000	Train	0.91	0.82
		Test	0.89	0.78
Hypertuned RF + class weight + Top **10** features	'max_features': 4, 'min_samples_split': 50, 'n_estimators': 1000	Train	0.92	0.85
		Test	0.91	0.82

Future studies, exploring deep learning or neural network algorithms, as well as ensemble methods that combine multiple machine learning models, could potentially improve performance by leveraging their ability to capture complex patterns and interactions in the data. These strategies could lead to a more balanced and robust prediction of canceled reservations.

Conclusions

Predicting reservation cancellations is crucial to hotels as it allows them to minimize revenue losses and optimize inventory. Using various machine learning models, we sought to predict hotel reservation cancellations with higher precision and recall than previous studies. We constructed multiple random forests and XGBoost models and tested different ways of processing imbalanced datasets. Of the two methods used to process our imbalanced dataset, the class_weight parameter and SMOTE, class_weight resulted in better performances.

Our best model was a hyperparameter-tuned random forest with class_weight set to "balanced," max_features of 14, min_samples_split of 50, and n_estimators of 1000. This model achieved training f1-scores of 0.93 for not canceled bookings and 0.86 for canceled bookings. Furthermore, it achieved

testing f1-scores of 0.92 for not canceled bookings and 0.83 for canceled bookings. Subsequently, we performed SHAP analysis to understand the contributions each input feature had on our model's predictions.

Choosing the top four and top ten contributing features, we trained and hyperparameter-tuned new random forest models on these features and compared their performances to our best model. The model trained on the top four features achieved train f1-scores of 0.91 for not canceled and 0.82 for canceled bookings. On testing data, it scored 0.89 for not canceled and 0.79 for canceled reservations. We consider this model's maximum 5% difference when compared to our best model to be optimum, given it was trained on only four features. The model trained on the top ten features achieved train and test f1-scores of 0.92 and 0.91, respectively, for not canceled bookings, and 0.85 and 0.82, respectively, for canceled reservations. This model had a 1% difference for all its predictions when compared to our best model, showing that only the top ten features in our data were necessary to achieve competitive results. In summary, this study found that hotel reservation cancellation prediction can be achieved with higher precision and recall than existing models. Not every feature in the dataset is required to achieve competitive optimum performance.

Data repository

https://github.com/GrLau/Predicting-Hotel-Res-Cancellations.git

References

1. Deo, R. C. (2015). Machine Learning in Medicine. *Circulation*, 132(20), 1920–1930. https://doi.org/10.1161/circulationaha.115.001593

2. Ghoddusi, H., Creamer, G. G., Rafizadeh, N. (2019). Machine learning in energy economics and finance: A review. *Energy Economics*, 81, 709–727. https://doi.org/10.1016/j.eneco.2019.05.006

3. Antonio, N., de Almeida, A., Nunes, L. (2017). Predicting Hotel Bookings Cancellation with a Machine Learning Classification Model. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. https://doi.org/10.1109/icmla.2017.00-11

4. Chen, S., Ngai, E. W. T., Ku, Y., Xu, Z., Gou, X., Zhang, C. (2023). Prediction of hotel booking cancellations: Integration of machine learning and probability model based on interpretable feature interaction. *Decision Support Systems*, 113959. https://doi.org/10.1016/j.dss.2023.113959

5. Raza, A. (n.d.). *Hotel Reservations Dataset*. www.kaggle.com. https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset

6. Hastie, T., Tibshirani, R., Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction.* 2nd ed. Springer, New York, NY.

7. Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2), 215–232.

8. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/a:1010933404324

9. Webb, G. I., Fürnkranz, J., Fürnkranz, J., Fürnkranz, J., Hinton, G., Sammut, C., Sander, J. et al. (2011). Decision Tree. *Encyclopedia of Machine Learning*, 263–267. https://doi.org/10.1007/978-0-387-30164-8_204

10. Chen, T., Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '*16, 785–794. https://doi.org/10.1145/2939672.2939785

11. *GridSearchCV*. (2025). Scikit-Learn. https://scikit-learn.org/1.6/modules/generated/sklearn.model_selection.GridSearchCV.html

12. *sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation*. (2018). Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

13. Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16(16), 321–357. https://doi.org/10.1613/jair.953

14. Orellana, E. (2020), December 10). *SMOTE - Emilia Orellana - Medium*. Medium. https://emilia-orellana44.medium.com/smote-2acd5dd09948

15. Lundberg, S., Lee, S.-I. (2017). *A Unified Approach to Interpreting Model Predictions*. ArXiv.org. https://doi.org/10.48550/arXiv.1705.07874